Predict Effect of Trump's Tweets on Stock Price Milestone

Tong Yang (tongy), Yuxin Yang (yuxiny)

December 16, 2017

1 Problem Description & Motivation

Many factors can influence market movement nowadays. With the power of social media, the market reacts to news events almost instantly. Reacting too slowly for a certain event might lead to loss of profiting opportunities or even negative PNL (Profit And Loss). In order to maintain profitability, one cannot merely rely on human labor in understanding the impact of the event. Natural Language Processing gives us competitive edge in understanding impacts of news events.

In this project, we would like to understand the impact and implication of a very specific news event release - President Trump's Tweets. Since one of our team member worked in algorithmic trading industry before, she is aware that there are many companies are actively using Twitter messages to trade. And those companies trade in large quantities that might actually lead to market index changes. In this project, we are going to assume that there are people actively trade with Trump's Twitter, and we want to predict how people react to Trump's Twitter releases. We want to understand how Trumps's Tweets can impact trader's behaviors and also the market.

The input to our models is vectors of words, with each vector representing one tweet's content. We then tried three different models: Naive Bayes, SVM and LSTM to predict market movement(rise/remain the same/drop) over certain time window after President Trump releases new Twitter messages. In addition, we also tested on different time-frames (ranging from 1 minute to 21 minutes) to experiment on the circulation time of the tweets.

2 Dataset

2.1 Raw Data

Our dataset includes three parts. The first part of our dataset is the training samples, which is the Trump's Tweets Archive. We found all of Trump's Tweets from May 4th, 2009 to Oct 20th, 2017 on `www.trumptwitterarchive.com`. [1] We exported their archive, and got a 5.9 MB csv file with $11,330$ tweets. The second part of our dataset is the outputs (labels). We are going to use $\sim$ 1GB of S&P 500 Index and underlying stocks minute resolution trading prices as an indication of market movements. We acquired this data source from Wharton Research Data Services (WRDS) [2], which is a matrix of dimension 146640×486, where $146,640$ refers to the time ticks from the beginning of 2016 up to today.The third part of our data is the pretrained GloVe[3] for word embedding, which will be used in the LSTM model. Using the embedding, each word can be representated as a vector in $\mathbb{R}^n$, where $n \in \{50, 100, 200, 300\}$.

2.2 Data Processing

Data processing is divided into three steps. The first step is to put training data together with the correct labels. In our case, the training data is Trump's Tweets, and the labels are the price changes. We firstly used Pandas Data Frame grouping methods to map the millisecond S&P 500 Index prices to minute resolution. Next, we calculated the percentage of change for each minute time stamp in the next minute. For example at 11:00 am, the price is $1,000$, and at 11:01 am, the price is $1,010$. The percentage of change is just $+1\%$ for 11:00 am. Then, we labeled each tweet with the price change after t minutes, where $t = [1, 21]$. After dropping tweets released during market closed, we got 11330 data samples that are overlapping with Trump's Tweets.

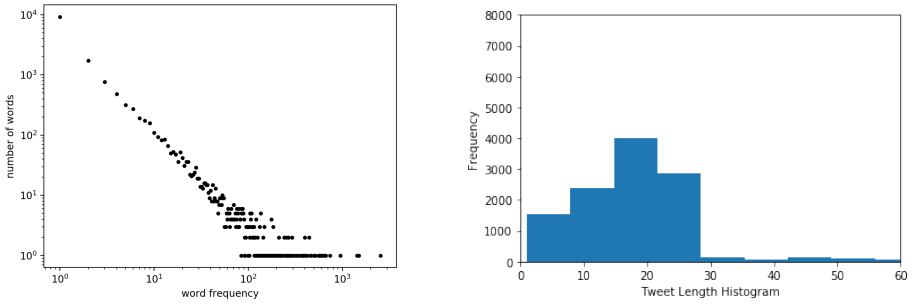
The second step is to prepare input data for Naive Bayes and SVM, which involves removing stop words, punctuations, and unnecessary numbers. For the purpose of this experiment, we decided to use Natural

Language Tool Kit (NLTK) from SKLearn library to help with the process. We removed stop words, did some text cleaning and vectorized the final text. Some sample entries look like below (csv format):

```
2017-10-20 11:31:00,"['report', 'united', 'kingdom', 'crime', 'rise',
'13', 'annually', 'amid', 'spread', 'radical', 'islamic', 'terror',
'good', 'must', 'keep', 'america', 'safe']",0.13643

2017-10-20 19:50:00,"['today', 'honor', 'unsecretary', 'general',
'antonioguterres', 'whitehouse', 'speak', 'u', 'appreciate']",-0.03889
```

Fig.1(a) shows the frequency of the words appeared in all the tweets. From the distribution it can be observed that many words appeared rarely, and some of them appeared very frequently. To be specific, there are in total $14,927$ distinct words, among which $9,242$ (62%) words appeared only once, $1,739$ (11.6%) words appeared twice, and 3 words appeared more than $1,000$ times. During modeling, we will focus more on the medium frequency words since they are likely to carry more classification information. Fig.1(b) shows the number of words after stop words removed for our baseline model. After finding out that most of the Tweets are less than 20 words. We used 20 dimensional data entries, and filled the unknown words as 399999, which is the last word in GloVe word embedding matrix representing not found words.



(a) Frequency of words in log-log scale

(b) Word2Vec Embedding

Figure 1: Data Statistics

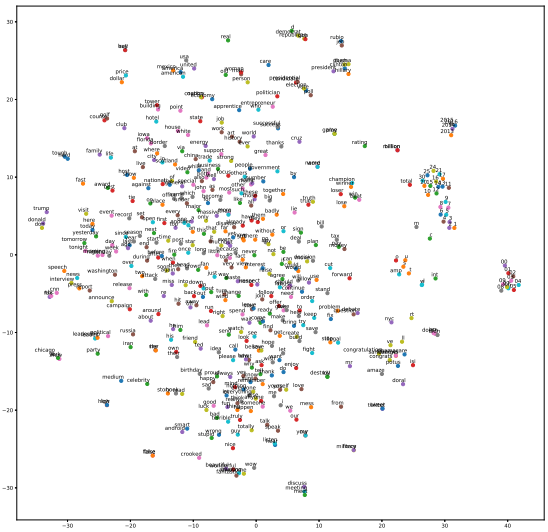


Figure 2: Word2Vec Embedding

The third part is to prepare input for LSTM. From Fig.1(c) it can be seen that the average length of each tweet is around 20, and more than 95% of the selected tweets are within length 30. We fixed the length

for LSTM input length to be 30 and represent each word with the pre-trained embedding vector of GloVe. Glove is a standard Word2Vec model, which uses the co-occurrence information of the words to map one-hot vector representation of words to lower dimension. Embedding dimensions of $50, 100, 200, 300$ are tried out during the experiment. Fig.2 shows the graph representation of embedding of words onto $\mathbb{R}^{300}$ after processed by PCA. Intuitively, words that have vectors close to each other are similar. As can be seen from the graph, years $(2013 - 2016)$ are close; "Cliton", "Obama", "Hillary", and "President" are close; "Dollar" and "Price" are close; "USA", "American" and "America" are close, etc.

3 Methods

The two baseline methods we used for predicting market price changing trend are SVM (support vector machine) and Naive Bayes. Both of them make predictions based on fixed time interval Δt. We form the tweets as input x and the change in market as output y with rising in stock labeled 1, no price changing labeled 0 and dropping labeled -1. These are all vectorized to a one-hot scale during label pre-processing, $i.e. Y \in \mathbb{R}^{9660\times 3}$.

We firstly chose Stochastic Gradient Descent classifier based on the assumption that each word carries a specific weight for the price to fluctuate up or down. And we also assume a linear relationship between the presence of each word and the price fluctuation probability. We used sk-learn SGD Classification model following stochastic gradient descent rule with hinge loss, which forms a linear SVM model for each of the three labels. [4]

$$\hat{w}_t = \hat{w}_{t-1} - \eta_t S^{-1}(\lambda \hat{w}_{t-1} + \phi_1'(w_{t-1}^T X_T, Y_t)X_t), \tag{1}$$

where η is the learning rate, w is the weight, X is the training matrix, and Y is the label matrix. ϕ is our loss function. Some preliminary results of this method are presented in the experiment section.

We also implemented Naive Bayes Gaussian for multinomial model, where we try to maximize the joint likelihood of the data $\prod_{i=1}^{m} p(x^{(i)}, y^{(i)})$. Here y^i takes value from $\{-1, 0, 1\}$ and $x^{(i)} \in \mathbb{R}^n$, where n is the size of vocabulary. Because of the word frequency distribution mentioned in the previous section, here we only select the medium frequency words, therefore n is a tuning parameter in our case and the value of which is smaller than 14927, which is the total number of distinct words in all tweets.

The third approach we implemented is LSTM Networks (Long Short Term Memory Network), one type of RNN (Recurrent Neural Network). Firstly we obtained vector representation of single tweet using GloVe (Global Vectors of Word Representation)[8]. The process transforms each word into a vector of $\mathbf{R}^n$, where $n \in \{50, 100, 200, 300\}$, which will be served as input x_t to RNN. The output y_t is a two dimensional Softmax function to predict the average S&P 500 yield in the next minute and the value of the output reflects the certainty of the output we have [9]. Since LSTM can better handle semantic analysis by considering the relation of words in the sentences, it is believed that LSTM would give better prediction of the stock price.

4 Results and Discussion

4.1 SVM model and Naive Bayes

The dataset was divided into training data, validation data, and test data. We used the last 2000 data points as the test data, and the rest was divided into 80% training data and 20% validation data. We implemented cross validation technique to randomize our training samples and reduce over-fitting. The accuracy of our project is defined by:

$$\text{accuracy} = \frac{\text{number of correctly predicted labels}}{\text{total amount of labels}} \tag{2}$$

Among all the time stamps after the tweets release, we found 5 minute labels to be most accurately predicted, and the performance tends to be stable with different cross validation development sets. In the Naive Bayes method, the accuracy is stably at the level of $\mathbf{42.5}\%$ on test sets. For SVM, we ran $1,000$ epochs for each cross validation set, and we ran 10 iterations of $1,000$ epochs. With this baseline model, we achieved $\mathbf{46.5}\%$ accuracy on test set prediction. As we later observed that the amount of positive and negative labels both takes up around 40% of the dataset, and the rest 20% are zero (no-change) labels, both model beat our 40% baseline, if we predict all labels to be positive or negative.

To help us better understand what our model has learned, we printed out the most informative words for determining the drop and rise of the market[10]. And the top 40 words for up and down labels are both

15.3715	division	-14.8110	imagination
14.0969	worst	-14.2299	james
13.9708	stupidly	-13.6232	reporter
13.6392	candidacy	-13.4340	puttster71
12.8453	jobless	-12.7844	diligence
12.5191	unaffordable	-12.5848	leibowitz
12.1724	96	-11.8904	rest
11.8494	lawrence	-11.8467	nhgop
11.8477	joelivan2	-11.6939	laura
11.7986	spectacle	-11.5271	understand
11.4673	disgusting	-11.3132	induct
11.4451	storm	-11.2103	refer
11.4271	misstetu	-10.8856	newspaper
11.4171	teacher	-10.5132	turnout
11.3710	bet22325450ste	-10.5052	skylerdeckard
11.1441	thehill	-10.4223	williebosshog
11.0000	brainpower	-10.4075	codyalliecats
10.9364	rave	-10.2830	core
10.8221	dude	-10.2624	brother
10.7374	2nd	-10.2522	ursulacurtin

(a) Most Indictaive Words For Negative Market Returns

14.3487	joenbc	-13.1766	division
12.8589	laura	-12.7904	accept
12.3803	leibowitz	-12.7456	arseniohall
12.2734	accrue	-12.7360	teacher
12.2476	lesleyclark	-11.9163	rolling
12.0598	politicalwire	-11.6819	disgusting
11.5967	impact	-10.9578	96
11.5886	skylerdeckard	-10.7496	thehill
11.4413	treated	-10.6875	idle
11.4013	wasted	-10.5661	drink
11.3998	divide	-10.4486	38000
11.2942	speculation	-10.2812	advice
11.2124	james	-10.1755	unique
11.1753	publicity	-10.1024	oct
11.0744	graphic	-10.0409	church
10.9842	ursulacurtin	-10.0392	assault
10.8414	watched	-10.0255	ryalsflair
10.8059	filing	-10.0247	20k
10.5621	slow	-9.9343	santa
10.5153	billingsley29	-9.9255	pleasurable

(b) Most Indictaive Words For Positive Market Returns

listed in the figure below. For negative change of the market, the word "division", "worst", "stupidly", "candidacy", "jobless" are the top five words that led to market drop, while words like "ursulacurtin", "brother", "core" are the ones that likely to have the opposite impact of market price drop.

Interestingly, for positive market change most influential words, we see a great symmetry between positive market change and negative market change. Words like "division" is likely to cause the exactly opposite of market rise. It's also very interesting to observe words like "JoeNBC", which is another Twitter account featuring Joe Scarborough, can positively impact the market whenever he tweets something that gets re-Tweeted by Trump. We can also see, that Trump's re-Tweets leads to most of the market rise, but Trump's own Tweets are more likely to cause negative impact. Noticeably, word "James" refers to the Iraq journalist and national hero James Wright Foley, who was brutally beheaded by ISIL. Intuitively, the mention of his name by Trump increased national unity and market's confidence. Besides that, we also noticed something very counter intuitive. In our database, the word "great" came up $1,431$ times, which is significantly higher than the occurrence of other words. About every $1/9$ times, Trump tweets "great" in his Twitter. And the fact that "great" as a word has almost zero impact on market change gives us some insights of how this word is probably used too much by our president, and carries less gravity than words indicating negative meanings like "worst" (which is the second most important indicator of market drop).

4.2 Recurrent Neural Network(LSTM)

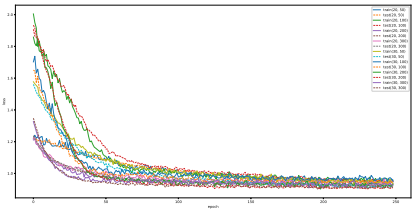
For the rest of our project, our main focus shifted to building a RNN model for prediction. Since our original price changes have accurate fluctuate values besides the binary rise and drop, we tried both softmax cross entropy loss and L2 loss during training. The intuition behind is that a rise of 3.5 is more significant than a rise of 0.01. Through experiment we found that softmax outperforms L2 loss, so we omit to present the second model in this paper, but we would advice anyone who plan to directly predict market return to modify the loss function to penalize more heavily on wrongly predicted labels, instead of simply the differences between labels and predictions.

The hyper-parameters of the model include Word2Vec embedding dimension size ($\{50, 100, 200, 300\}$), length of LSTM units (fixed tweet length window)($\{20, 30\}$), and number of LSTM states($\{5, 32, 64\}$). In order to better interpret our result, and to decide which time window to use in order to make the best predictions, we produced the following graphs. Fig.3 shows the plot of loss and accuracy of model with different parameters after different epochs. By analyzing the result, we found the following facts:

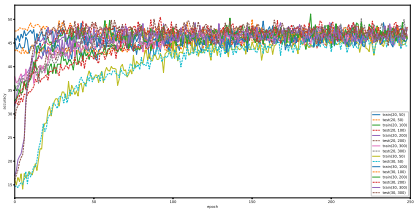
(a) Higher embedding dimension gives better prediction: $\mathbb{R}^{50}$ performs much worse than $\mathbb{R}^{200}$, while $\mathbb{R}^{200}$ and $\mathbb{R}^{300}$ gives similar performance;

(b) Longer tweet length window results in higher prediction accuracy: setting the truncate length of tweet to 30 rather than 20 results in higher accuracy because it includes more text information in the training example;

(c) Smaller LSTM state gives better performance: setting the state number to 5 gives the lowest loss. When the state number is set to 64, we found significant overfitting of the data with training accuracy reaching 95% and dev accuracy 39%.



(c) Model Loss with different parameters vs. number of iterations



(d) Model Accuracy with different parameters vs. number of iterations

Figure 3: LSTM Model Loss and Accuracy for tuning hyper-parameters. The graph on the left plots the loss of train_set (solid line) and test_set (dotted line) over number of epochs our model had run. In legend, numbers (50, 100, 200 or 300) represents the dimensions of word embeddings. And the graph on the right plots the accuracy of each model over number of iterations.

To summarize, after lots of hyper parameter tuning, and implementing different ways of NLP data parsing. Our final LSTM model with Tweet length 30, stop words included, 64 states in LSTM units, batch_size 24, and 300 dimension word embedding is the best performing one among all what we have tried. And this best performing model produces a final test set accuracy of 48.3%, beating human prediction by 5.3%, and beating best performing SVM model by 1.8%. The model converges after around 50 epochs.

4.3 Model Comparison and Analysis

Comparing the prediction result on the final test set data, We found linear SVM to have better performance than Naive Bayes model. This might be due to that Naive Bayes assumes independent conditional probability, which might not be true for our case. Some words appear more frequently than others. And SVM is more immune to this short-come. We also found that LSTM model has the best accuracy. This is exactly what we expected, because LSTM not only extracts features from different words (for instance "great" may be a key word in the price rise), but also preserves the timing information of a piece of time series data. We used Tensorflow LSTM dynamic cells to automatically unroll the time series into information that contains the timing sequence information. Therefore, it does the semantic analysis based on the whole tweet content, instead of looking at each word individually. For instance when "great" appears together with some privative prefix, it would more likely to lead to the drop of price. Thus, LSTM should be the most robust among all of the models we implemented, and this is exactly what we observed.

5 Conclusion and Future Steps

To summarize, this has been an extremely fun dataset to work with. We can conclude that predicting President Trump's Tweets does have the potential to be profit generating, because by deep learning, we were able to beat human prediction by more than 5%, and reached an accuracy of 48% in three class classification problem. However, only Trump's Tweets seem to be not quite sufficient for us to make a deterministic prediction. There are a large amount of changes of the market that we were not able to fit our limited features with. We would advice anyone who want to explore more on trading with NLP to implement more features in the future, including Tweets by some other politicians or celebrities, or news headline releases as well.

We manually put together this original dataset, in order to explore the potentials of NLP in the realm of algorithmic trading, and also to understand politically how much weight each of President Trump's Tweet carries. The dataset size is not too small to cause over-fitting, yet not too large that requires excessive amount of computing. Not only did we observe that our baseline can produce some very intuitive results, we also observed a significant improvement in accuracy after switching to deep learning. We are overall very pleased with what we learned.We give great thanks to the wonderful instructors and TAs for CS229 for providing great help over this short time frame for us to go this far with our project. And we definitely plan to push the potential of our dataset further by publicly publishing it on Kaggle or some private GIT platform to let other people try it in the future.

Group Contributions

Tong and Yuxin divided the work evenly, and both have had lots of fun putting this project together.

References

[1] “Trump Twitter Archive.” Trump Twitter Archive, www.trumptwitterarchive.com/.

[2] Wharton Research Data Services. , 1993. Internet resource.

[3] Pennington, Jeffrey. GloVe: Global Vectors for Word Representation, nlp.stanford.edu/projects/glove/.

[4] Zhang, Tong. "Solving large scale linear prediction problems using stochastic gradient descent algorithms." Proceedings of the twenty-first international conference on Machine learning. ACM, 2004.

[5] C.D. Manning, P. Raghavan and H. Schütze (2008). Introduction to Information Retrieval. Cambridge University Press, pp. 234-265.

[6] V. Metsis, I. Androutsopoulos and G. Paliouras (2006). Spam filtering with Naive Bayes – Which Naive Bayes? 3rd Conf. on Email and Anti-Spam (CEAS).

[7] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9.8 (1997): 1735-1780.

[8] “Vector Representations of Words | TensorFlow.” TensorFlow, www.tensorflow.org/tutorials/word2vec.

[9] Deshpande, Adit. “Perform Sentiment Analysis with LSTMs, Using TensorFlow.” O’Reilly Media, 13 July 2017, www.oreilly.com/learning/perform-sentiment-analysis-with-lstms-using-tensorflow.

[10] “Libelli.” Text Classification with NLTK and Scikit-Learn · Libelli, bbengfort.github.io/tutorials/2016/05/19/text-classification-nltk-sckit-learn.html.